CLAIMS

1. A method of scanning a computer file for virus infections comprising:

a) identifying program code within the file

b) identifying the compiler used to create the program code

c) determining the frequency distribution of selected machine code instructions or sequences of such instructions; and

d) flagging the file as possibly infected with a virus, or not, on the basis of comparison of the determined frequency distribution with a frequency distribution of machine code instructions or sequences thereof expected for that compiler.

2. A method according to claim 1 wherein step c) comprises the step, working from an entry point of the program, of

b1) tracing an execution graph by decoding successive instruction opcodes and updating frequency counts of decoded instructions as this tracing proceeds.

3. A method according to claim 2 wherein when, during step b1), a subroutine call or conditional branch instruction is encountered, the destination of the call or branch instruction is pushed onto a stack, tracing proceeds into the subroutine call, and when a return instruction is encountered, the pushed location is popped from the stack and tracing continues with the following instructions, if any.

4. A method according to claim 1, 2 or 3, wherein the program code is examined for opcode constructs, such as subroutine-call and subroutine-return, instruction sequences which are expected to occur a known ratio to each other and, if the ratio actually found differs from the known one by more than a certain amount, the file is flagged as possibly viral, or subject to further processing.

5. A method according to any one of the preceding claims and including the step, where step d) flags the file as possibly viral, of comparing the program code with a list of permissible exceptions and suppressing the flag if the program code is considered to be in the exception list.

6. A system for scanning a computer file for virus infections comprising:

a) means for identifying program code within the file

b) means for identifying the compiler used to create the program code

c) means for determining the frequency distribution of selected machine code instructions or sequences of such instructions; and

d) means for flagging the file as possibly infected with a virus, or not, on the basis of comparison of the determined frequency distribution with a frequency distribution of machine code instructions or sequences thereof expected for that compiler.

7. A system according to claim 6 the frequency determining means c) includes tracing means, the tracing means being operable, working from an entry point of the program, to trace an execution graph by decoding successive instruction opcodes and updating frequency counts of decoded instructions as this tracing proceeds.

8. A system according to claim 7 wherein the tracing means is operable such that when a subroutine call or conditional branch instruction is encountered, the destination of the call or branch instruction is pushed onto a stack, tracing proceeds into the subroutine call, and when a return instruction is encountered, the pushed location is popped from the stack and tracing continues with the following instructions, if any.

9. A system according to claim 6. 7 or 8, and including means for examining the program code for opcode constructs,. such as subroutine-call and subroutine-return, instruction sequences which are expected to occur a known ratio to each other and, if the ratio actually found differs from the known one by more than a certain amount, the file is flagged as possibly viral, or subject to further processing

10. A system according to any one claims 6 to 9 and including means, operable when the means d) flags the file as possibly viral, to compare the program code with a list of permissible exceptions and suppressing the flag if the program code is considered to be in the exception list.